Tecnica del junction tree per l'inferenza probabilistica nelle reti Bayesiane

Niccolò Menghini

9 febbraio 2023

1 Introduzione

In questo elaborato si presenta il metodo di costruzione di un junction tree per una rete Bayesiana, rappresentata da un DAG, descritto in dettaglio nel capitolo 4 del libro di Jensen *Introduction to Bayesian Networks* [Jen96]. Il metodo si compone di vari passaggi che implementano diversi algoritmi per effettuare varie operazioni sulla rete di partenza (come ad esempio la triangolazione di un grafo, o l'individuazione di uno spanning tree etc.) manipolandola fino ad ottenere un junction tree su cui sarà poi possibile implementare algoritmi migliori per inserire prove e aggiornare le probabilità. I junction tree ottenuti possono essere poi confrontati con i junction tree proposti dal programma Hugin Lite, che implementa lo stesso metodo di costruzione degli alberi, per verificare la bontà del risultato.

2 Procedura di costruzione dei junction tree

Gli step principali della procedura sono:

- Costruire un ***moral graph*** della rete di partenza
- ***Triangolare*** il moral graph
- Costruire il ***junction graph*** del grafo triangolato
- Trovare il ***maximum weight spanning tree*** del junction graph

Quest'ultimo passaggio ci fornisce uno dei junction tree relativi alla rete Bayesiana di partenza.

Tutti gli algoritmi che verranno successivamente citati sono applicati a una struttura per modellazione di grafi creata ad hoc, inoltre al termine di ogni passaggio sono generati dei file in formato PDF che mostrano una rappresentazione grafica di ciò che è stato ottenuto da una particolare trasformazione. Per fare ciò è stata utilizzata la libreria GraphViz per python.

2.1 Costruzione del Moral Graph

Per costruire il moral graph di un DAG è necessario semplicemente connettere tra di loro i nodi che hanno un nodo figlio in comune. Una volta fatto questo il programma costruisce un grafo non orientato che calca esattamente il DAG di partenza con l'aggiunta degli archi generati da questo passo di moralizzazione.

2.2 Triangolazione del moral graph

Esistono vari algoritmi per la triangolazione di un grafo, in questo programma si è scelto di implementarne due per vedere se, ed eventualmente come, una differente triangolazione del grafo impatta il risultato finale.

Il primo algoritmo implementato è stato ripreso dall'articolo di Berry, Blair, Heggernes & Peyton [Ber+04] e da loro denominato ***MCS-M***. Il grafo risultante è triangolato come dimostrato nell'articolo.

Il secondo dei due riprende alcune considerazioni fatte sul **teorema 4.9** presente nel libro di Jensen e viene chiamato comunemente ***Elimination Game***: l'algoritmo lavora eliminando tutti i nodi *in un qualche ordine* e aggiungendo archi in modo tale da collegare tutti i vicini di un nodo eliminato a due a due (se lo sono già, non si aggiungono archi). Il grafo risultante è triangolato per il teorema sopra citato.

2.3 Costruzione del junction graph

Una volta triangolato il grafo, si passa alla costruzione del junction graph da cui poi si deriva il junction tree. A questo fine è stato implementato l'algoritmo di Bron-Kerbosch [BK73], nella sua versione senza pivot, che permette di elencare tutte le cricche massimali del grafo. Le cricche trovate andranno a rappresentare i nodi nel junction graph, gli archi invece sono modellati secondo quelle che sono le intersezioni tra le varie cricche: ogni arco avrà infatti a lui assegnata un'etichetta che contiene le variabili in comune tra due cricche.
Il programma si occupa della costruzione del junction graph secondo la modalità appena descritta.

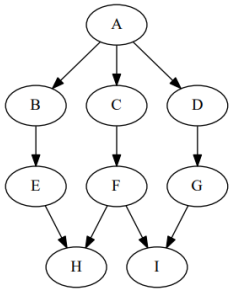
2.4 Trovare il junction tree

Poiché il junction tree non è altro che uno spanning tree di peso massimo, viene implementato nel programma l'algoritmo di Kruskal modificato al fine di aggiungere per primi gli archi con peso maggiore invece di quelli con peso minore. Il peso degli archi viene definito in questo caso come il numero di variabili presenti nell'etichetta dell'arco.
Al termine di questo passaggio il programma ha con successo costruito un junction tree della rete Bayesiana di partenza e termina la sua esecuzione.

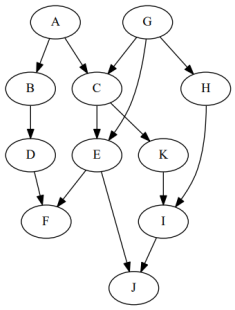
3 Esperimenti e risultati

3.1 Esperimenti eseguiti

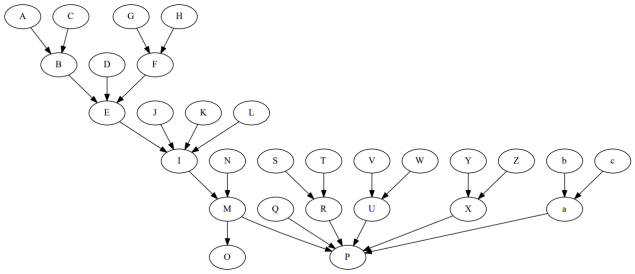
Per testare la correttezza del programma sono stati richiesti 3 test, nello specifico un test su una rete presente nel libro di Jensen (**figura 4.18**), un test su una rete piccola (circa 10 variabili) e un test su una rete media (circa 30 variabili); oltre a questi sono stati effettuati altrettanti test per confrontare il secondo algoritmo di triangolazione implementato e vari altri su reti di varia dimensione.



(a) DAG relativo alla rete test



(b) DAG relativo alla rete piccola

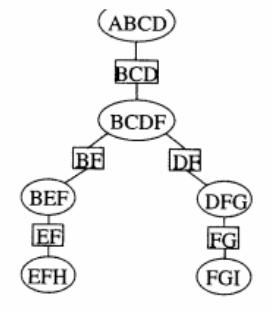


(c) DAG relativo alla rete media

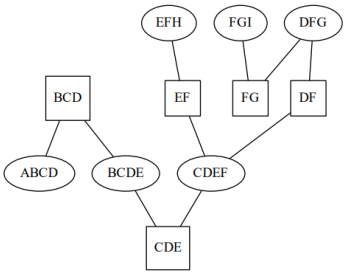
Figura 1: I DAG relativi alle reti Bayesiane su cui è stato testato l'algoritmo

3.2 Risultati e osservazioni

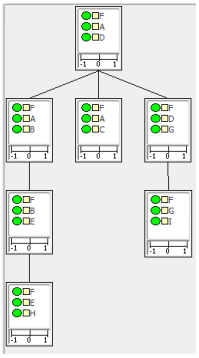
Dai risultati ottenuti si nota (anche solo osservando i junction tree relativi alla prima rete Bayesiana in Fig. 2) che difficilmente i junction tree generati dai differenti programmi, pur seguendo lo stesso metodo di costruzione, coincidano l'uno con l'altro.



(a) Junction tree come riportato in figura 4.18 sul libro di Jensen



(b) Junction tree generato dall'algoritmo



(c) Junction tree generato da Hugins

Figura 2: I vari risultati degli esperimenti fatti sulla prima rete

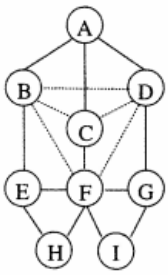
Partiamo analizzando i risultati ottenuti dagli esperimenti fatti sulla prima rete. È subito possibile notare come i tre alberi non coincidano, se non in pochi punti, nonostante il metodo utilizzato sia lo stesso. La differenza tra il risultato riportato libro di Jensen (Fig. 2a) e quello generato dal programma Hugins (Fig. 2c) non era attesa, tuttavia è servita per focalizzare alcuni concetti che giustificano l'eterogeneità dei risultati.

Sono due gli step dell'algoritmo che possono portare a delle differenze: il passo di triangolazione e il passo di costruzione dello spanning tree.

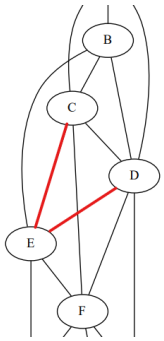
In base a quella che è la definizione di triangolazione si può notare come sia un processo piuttosto arbitrario: è sufficiente l'esistenza di un ordine di eliminazione che non aggiunga nuovi archi per dire che un grafo è triangolato. Sul libro di Jensen viene mostrato un esempio di come siano possibili più triangolazioni di uno stesso grafo, di seguito la fig. 3 mostra come la triangolazione effettuata dall'algoritmo implementato differisca da quella effettuata sulla stessa rete nel libro.

Per quanto concerne la costruzione dello spanning tree, la sua unicità è garantita solo nel caso in cui tutti i pesi siano differenti. Spesso non è questo il caso e si può quindi differenziare la costruzione rispettando la definizione, come ad esempio in figura 4b, dove il cluster 'FAC' poteva essere equivalentemente connesso al cluster 'FAB' attraverso un ramo di peso 2, stesso peso della connessione che invece è stata fatta con il cluster 'FAD'.

Persistono invece in tutti i cluster che derivano da cricche massimali formatesi durante lo step di moralizzazione, in questo caso specifico i cluster 'EFH' e 'FGI'.



(a) Triangolazione della rete di test come riportata sul libro di Jensen

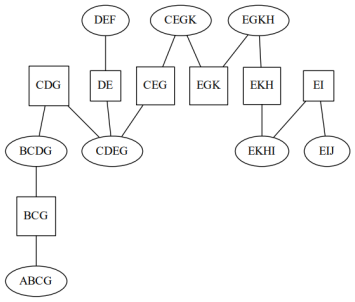


(b) Particolare della triangolazione effettuata dall'algoritmo

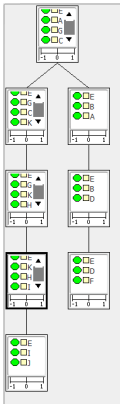
Figura 3: Confronto tra le triangolazioni effettuate dall'algoritmo e da Jensen, in rosso gli archi aggiunti dall'algoritmo che differiscono da quelli aggiunti da Jensen

Successivamente (Fig. 4) vengono riportati i risultati ottenuti sulla rete piccola, dove si può notare una differenza ancora più spiccata tra le costruzioni dei due junction tree. In accordo con le affermazioni precedentemente fatte, una differente triangolazione fornisce un albero finale diverso, come si può vedere anche in figura 4c.

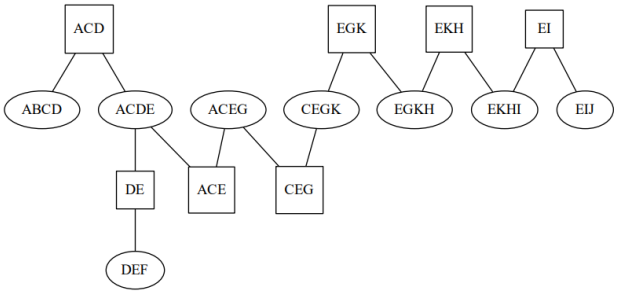
Infine si riportano i risultati ottenuti sulla rete media. La proprietà particolare di questa rete è che non ha nessun ciclo, motivo per il quale la triangolazione non ha effetto sull'albero finale. Questo si può osservare notando che i cluster del junction tree finale contengono le stesse variabili, qualunque sia l'algoritmo di triangolazione scelto. Sfortunatamente la versione di Hugins a mia disposizione non mi permette di costruire il junction tree di una rete così ampia, per cui non mi è possibile dimostrare visivamente questa affermazione se non fornendo i risultati dell'algoritmo in fig. 5.



(a) Junction tree generato dall'algoritmo

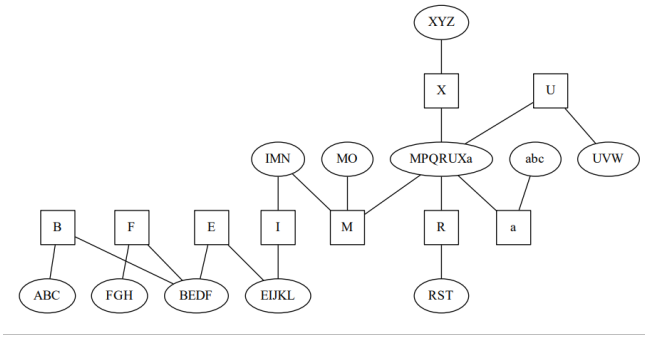


(b) Junction tree generato da Hugins

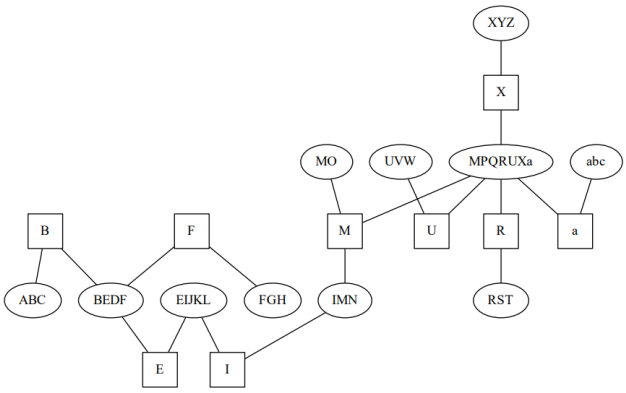


(c) Junction tree generato dall'algoritmo utilizzando un differente metodo per la triangolazione

Figura 4: I vari risultati degli esperimenti fatti sulla rete piccola



(a) Junction tree generato dall'algoritmo



(b) Junction tree generato dall'algoritmo utilizzando un differente metodo di triangolazione

Figura 5: I vari risultati degli esperimenti fatti sulla rete media

Riferimenti bibliografici

[Ber+04] Anne Berry et al. ≪Maximum Cardinality Search for Computing Minimal Triangulations of Graphs≫. In: *Algorithmica* 39 (2004), pp. 287–298.

[BK73] Coen Bron e Joep Kerbosch. ≪Algorithm 457: finding all cliques of an undirected graph≫. In: *Communications of the ACM* 16 (1973), pp. 575–577.

[Jen96] F. V. Jensen. *An introduction to Bayesian networks*. United Kingdom: Springer New York, 1996.